

## Annex A

(normative)

### (Initial draft of) EXPRESS data transfer model

This annex presents an EXPRESS model of both EXPRESS and EXPRESS-G according to the viewpoint of the exchange of data representing such models between CASE-like (Computer Aided Software Engineering) systems using transfer technology like that defined in ISO 10303-21. The model of EXPRESS is designed to enable the reconstruction of an original lexical EXPRESS source from the data instances. Similarly, the model of EXPRESS-G is designed to enable a simulcrum of an original EXPRESS-G graphical model to be created from the data instances.

#### NOTES

1 – The models presented in this annex are not semantic (i.e., meta-models) of either EXPRESS or EXPRESS-G. Neither are they syntactic models. Rather, they describe sufficient information to enable reconstruction of an original model.

2 – The models are heavily biased towards a data modeling style rather than an information modeling style. For example, several places in the model use the SELECT construct rather than subtyping in order to minimise the size of the actual data instances.

3 – The constraints embodied in the model are not sufficient to ensure that data instances will be reconstructable as conforming EXPRESS or EXPRESS-G. It is assumed that an EXPRESS or EXPRESS-G processor will be used to test for conformity. Adding the constraints necessary to ensure conformance will needlessly complicate the model with respect to its intended purpose.

4 – Unless otherwise stated, the examples of data instances in this annex use the EXPRESS-I syntax as defined in ISO 10303-12:1997.

The complete model consists of several schemas. Figure A.1 shows the organization of the schemas within the model.

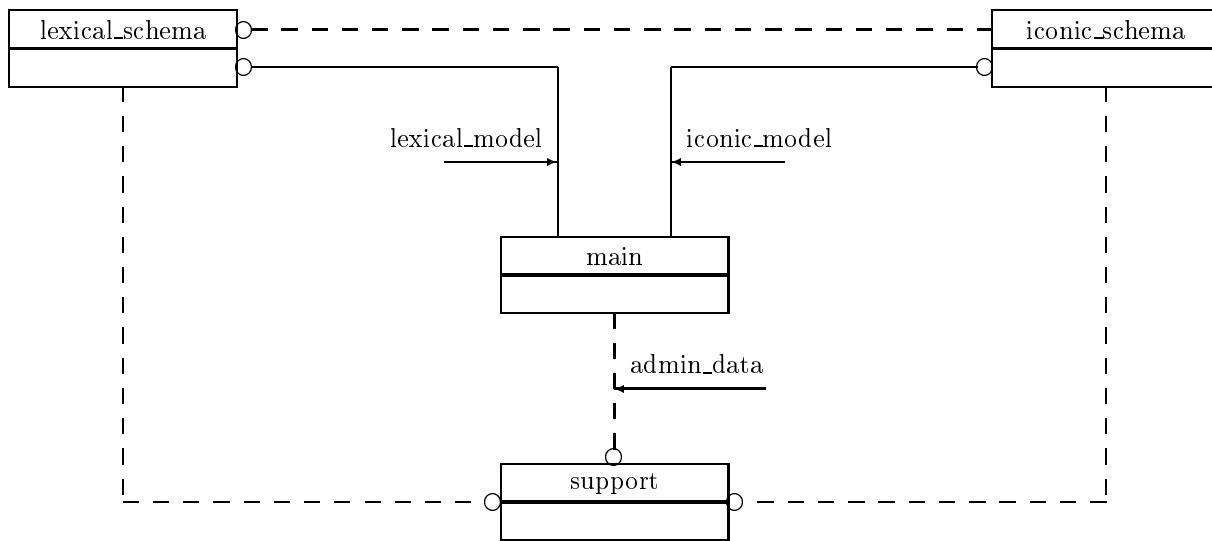
#### A.1 Schema main

This is the main schema for a data transfer model of EXPRESS-G models.

##### EXPRESS specification:

```
*)
SCHEMA main;

USE FROM lexical_schema (lexical_model);
USE FROM iconic_schema (iconic_model);
REFERENCE FROM support (admin_data);
(*
```



**Figure A.1 – Complete schema level diagram of a model for the transfer of EXPRESS-G model representations. (Page 1 of 1)**

## A.1.1 Entity definitions

### A.1.1.1 model\_views

A **model\_views** provides for the association of one or more **iconic\_model** with a **lexical\_model**. ■

EXPRESS specification:

```

*)
ENTITY model_views;
  lexical      : lexical_model;
  iconic       : SET [1:?] OF iconic_model;
  administrative : admin_data;
END_ENTITY;
(*
  
```

Attribute definitions:

**lexical:** An EXPRESS lexical model.

**iconic:** One or more EXPRESS-G iconic models that represent different views of the **lexical** model.

**administrative:** Adminstrative data.

EXAMPLE 1 – A data instance of a **model\_views** where two **iconic\_models** are associated with a **lexical\_model**.

EXPRESS-I specification:

```
mv = model_views{lexical -> @lm;
                 iconic -> (@im1, @im2);
                 administrative -> @admin;};
```

### A.1.2 End of schema main

EXPRESS specification:

```
*)
END_SCHEMA; -- end of main
(*)
```

## A.2 Schema Lexical\_Schema

This schema provides for the association of several SCHEMA and describes a SCHEMA sufficiently for the EXPRESS source code of the schema to be reconstructed from data instances.

EXPRESS specification:

```
*)
SCHEMA lexical_schema;
  REFERENCE FROM support;
(*)
```

### A.2.1 Type definitions

#### A.2.1.1 any\_type

All the kinds of EXPRESS types.

EXPRESS specification:

```
*)
TYPE any_type = SELECT(named_type, simple_type);
END_TYPE;
(*)
```

#### A.2.1.2 simple\_type

A representation of the EXPRESS simple types.

EXPRESS specification:

```
*)  
TYPE simple_type = ENUMERATION OF (Sbinary,  
                                     Sboolean,  
                                     Sinteger,  
                                     Slogical,  
                                     Snumber,  
                                     Sreal,  
                                     Sstring);  
END_TYPE;  
(*
```

### A.2.1.3 named\_type

The EXPRESS named types.

EXPRESS specification:

```
*)  
TYPE named_type = SELECT(Xentity, Xtype);  
END_TYPE;  
(*)
```

## A.2.2 Entity definitions

### A.2.2.1 lexical\_model

A **lexical\_model** is an information model specified using the EXPRESS lexical language. A **lexical\_model** is composed of one or more schema.

EXPRESS specification:

```
*)  
ENTITY lexical_model;  
  components : SET [1:?] OF Xschema;  
  administrative : admin_data;  
END_ENTITY;  
(*)
```

Attribute definitions:

**components:** The set of schema comprising the information model.

**administrative:** Adminstrative data.

EXAMPLE 2 – A data instance of a **lexical\_model**

EXPRESS-I specification:

```
lm = lexical_model{components -> (@sch1, @sch2, @sch3);
                  administrative -> @admin;};
```

**A.2.2.2 Xschema**

An **Xschema** is a representation of an EXPRESS SCHEMA.

EXPRESS specification:

```
*)
ENTITY Xschema;
  name      : express_id;
  uses      : SET OF Xuse;
  references : SET OF Xreference;
  constants  : SET OF Xconstant;
  types      : SET OF Xtype;
  entities   : SET OF Xentity;
  rules      : SET OF Xrule;
  functions  : SET OF Xfunction;
  procedures : SET OF Xprocedure;
END_ENTITY;
(*
```

Attribute definitions:

**name:** The identifier of the schema.

**uses:** The set of declarations used from other schemas.

**references:** The set of declarations referenced from other schemas.

**constants:** The set of constant declarations.

**types:** The set of type declarations.

**entities:** The set of entity declarations.

**rules:** The set of rule declarations.

**functions:** The set of function declarations.

**procedures:** The set of procedure declarations.

EXAMPLE 3 – A data instance of a **Xschema**

EXPRESS-I specification:

```
sch1 = Xschema{name -> 'my_schema';
```

```
uses -> ();
references -> (@refimport);
constants -> ();
types -> (@xtype1, @xtype2);
entities -> (@ent1, @ent2, @ent3);
rules -> ();
functions -> (@fun1);
procedures -> ()};
```

### A.2.2.3 Xuse

Declarations USED from another SCHEMA.

EXPRESS specification:

```
*)
ENTITY Xuse;
  source : Xschema;
  imports : SET OF perhaps_renamed;
END_ENTITY;
(*
```

Attribute definitions:

**source:** The SCHEMA used as the exporter.

**imports:** The imported declarations. If the set is empty, then all the ENTITY definitions are imported.

Informal propositions:

**ip1:** The imports are limited to ENTITY types.

EXAMPLE 4 – A data instance of an **Xuse** which represents the EXPRESS code:

```
USE FROM a_schema (entity_1,
  entity_2 AS renamed_entity_2);
```

EXPRESS-I specification:

```
xuse1 = Xuse{source -> @his_schema;
  imports -> (@prn1, @prn2);};
```

### A.2.2.4 Xreference

Declarations REFERENCED from another SCHEMA.

EXPRESS specification:

```
*)
ENTITY Xreference;
  source  : Xschema;
  imports : SET OF perhaps_renamed;
END_ENTITY;
(*)
```

Attribute definitions:

**source:** The SCHEMA used as the exporter.

**imports:** The imported declarations. If the set is empty, then all the definitions are imported.

EXAMPLE 5 – A data instance of an **Xreference** which represents the EXPRESS code:

```
REFERENCE FROM another_schema;
```

EXPRESS-I specification:

```
xref1 = Xreference{source -> @her_schema;
                     imports -> ()};
```

### A.2.2.5 perhaps\_renmamed

The association of a declaration with a name alias.

EXPRESS specification:

```
*)
ENTITY perhaps_renamed;
  original : named_type;
  aka      : OPTIONAL express_id;
DERIVE
  context_name : express_id := NVL(aka, named_type.name);
END_ENTITY;
(*)
```

Attribute definitions:

**original:** The original declaration.

**aka:** An optional alias for the **name** of the original.

**context\_name:** The alias, if it exists, otherwise the original name.

EXAMPLES Data instances of an **perhaps\_renamed**

6 – A representation of the EXPRESS code:

```
entity_1
```

EXPRESS-I specification:

```
prn1 = perhaps_renamed{original -> @ent1;
                      aka -> ?;
                      context_name <- 'entity_1';};
```

7 – A representation of the EXPRESS code:

```
entity_2 AS renamed_entity_2
```

EXPRESS-I specification:

```
prn2 = perhaps_renamed{original -> @ent2;
                      aka -> 'renamed_entity_2';
                      context_name <- 'renamed_entity_2';};
```

### A.2.2.6 Xconstant

A named constant.

EXPRESS specification:

```
*)
ENTITY Xconstant;
  name      : express_id;
  the_type  : type_spec;
  the_value : expression;
END_ENTITY;
(*)
```

Attribute definitions:

**name:** The name of the constant.

**type\_def:** The type of the constant.

**the\_value:** An expression which evaluates to the value of the constant.

EXAMPLE 8 – Data instances of an **Xconstant** corresponding to the EXPRESS code:

```
-- CONSTANT
  twopi  : REAL      := 2.0 * PI;
  origin : point_2d := point_2d(0.0, 0.0);
```

```
-- END_CONSTANT;
```

EXPRESS-I specification:

```
const1 = Xconstant{name -> 'twopi';
                    the_type -> @areal;
                    the_value -> ('2.0 * PI');};
const2 = Xconstant{name -> 'origin';
                    the_type -> @a2dpoint;
                    the_value -> ('point_2d(0.0, 0.0)');};
```

### A.2.2.7 Xtype

A representation of an EXPRESS TYPE.

EXPRESS specification:

```
*)
ENTITY Xtype
  ABSTRACT SUPERTYPE OF (ONEOF(Xenumeration, Xselect, defined_type));
  name      : express_id;
  constraints : where_clause;
END_ENTITY;
(*)
```

Attribute definitions:

**name:** The identifier of the type.

**constraints:** The constraints on the type.

### A.2.2.8 Xenumeration

A representation of an EXPRESS ENUMERATION type.

EXPRESS specification:

```
*)
ENTITY Xenumeration
  SUBTYPE OF (Xtype);
  ids : LIST OF UNIQUE express_id;
END_ENTITY;
(*)
```

Attribute definitions:

**ids:** The enumeration identifiers.

EXAMPLE 9 – A data instance of an **Xenumeration** corresponding to the EXPRESS code:

```
TYPE enum = ENUMERATION OF (first, second, third);
END_TYPE;
```

EXPRESS-I specification:

```
enum[1] = Xtype{name -> 'enum';
                 constraints -> ();
                 SUPOF(@2);};
enum[2] = Xenumeration{SUBOF(@1);
                       ids -> ('first', 'second', 'third');};
```

### A.2.2.9 Xselect

A representation of an EXPRESS SELECT type.

EXPRESS specification:

```
*)
ENTITY Xselect
  SUBTYPE OF (Xtype);
  selection : SET OF named_type;
END_ENTITY;
(*)
```

Attribute definitions:

**selection:** The set of named types forming the potential selection.

EXAMPLE 10 – A data instance of an **Xselect** corresponding to the EXPRESS code:

```
TYPE choose = SELECT (entity_1, enum);
END_TYPE;
```

EXPRESS-I specification:

```
sel[1] = Xtype{name -> 'choose';
                constraints -> ();
                SUPOF(@2);};
sel[2] = Xselect{SUBOF(@1);
                 selection -> (@ent1, @enum);};
```

### A.2.2.10 defined\_type

A representation of an EXPRESS defined type which is neither an ENUMERATION type nor a SELECT type.

EXPRESS specification:

```
*)  
ENTITY defined_type  
  SUBTYPE OF (Xtype);  
  type_def : type_spec;  
END_ENTITY;  
(*
```

Attribute definitions:

**type\_def:** The type.

EXAMPLE 11 – A data instance of a **defined\_type** corresponding to the EXPRESS code:

```
TYPE a_length = REAL;  
WHERE  
  wr1 : SELF >= 0.0;  
END_TYPE;
```

EXPRESS-I specification:

```
def[1] = Xtype{name -> 'a_length';  
               constraints -> ('wr1 : SELF >= 0.0;');  
               SUPOF(@2);};  
def[2] = defined_type{SUBOF(@1);  
                     type_def -> @areal;};
```

### A.2.2.11 type\_spec

A representation of an EXPRESS type specification.

EXPRESS specification:

```
*)  
ENTITY type_spec;  
  aggregation : LIST OF Saggregate;  
  base        : any_type;  
END_ENTITY;  
(*
```

Attribute definitions:

**aggregation:** An aggregation specification.

**base:** The base type.

EXAMPLES Data instances of a **type\_spec**

12 – A representation of the EXPRESS type specification code:

REAL

EXPRESS-I specification:

```
areal = type_spec{aggregation -> () ;
                  base -> !Sreal;};
```

13 – A representation of the EXPRESS type specification code:

BAG [1:?] OF INTEGER

EXPRESS-I specification:

```
bag_ip_integer = type_spec{aggregation -> (@bag_1p) ;
                           base -> !Sinteger;};
```

### A.2.2.12 Saggregate

A representation of an EXPRESS aggregate.

EXPRESS specification:

```
*)
ENTITY Saggregate
  ABSTRACT SUPERTYPE OF (ONEOF(Sarray, Sbag, Slist) ANDOR
                           ONEOF(Sarray, Sbag, Slist, unique_elements));
  bound : OPTIONAL bound_spec;
END_ENTITY;
(*)
```

Attribute definitions:

**bound:** The aggregate boundaries.

### A.2.2.13 bound\_spec

A representation of an aggregate boundary specification

#### EXPRESS specification:

```
*)  
ENTITY bound_spec;  
    low : expression;  
    high : expression;  
END_ENTITY;  
(*
```

#### Attribute definitions:

**low:** The lower boundary.

**high:** The higher boundary.

EXAMPLE 14 – Data instances of a **bound\_spec** representing the EXPRESS code fragments:

```
[1:?:]  
[1:3]
```

#### EXPRESS-I specification:

```
b1p = bound_spec{low -> ('1');  
                  high -> ('?');}  
b1to3 = bound_spec{low -> ('1');  
                  high -> ('3');}
```

### A.2.2.14 unique\_elements

An **Saggregate** in which the elements of the aggregate shall be unique.

#### EXPRESS specification:

```
*)  
ENTITY unique_elements  
    SUBTYPE OF (Saggregate);  
END_ENTITY;  
(*
```

### A.2.2.15 Sarray

A representation of an EXPRESS ARRAY.

EXPRESS specification:

```
*)  
ENTITY Sarray  
  SUBTYPE OF (Saggregate);  
  index_type : any_type;  
  index_from : expression;  
END_ENTITY;  
(*
```

Attribute definitions:

**index\_type:** The EXPRESS type of the ARRAY index.

**index\_from:** The starting value for the **index\_type**.

EXAMPLE 15 – A data instance of an **Sarray** corresponding to the EXPRESS code fragment:

```
ARRAY [1:3] INDEXED BY INTEGER OF OPTIONAL UNIQUE
```

EXPRESS-I specification:

```
a1to3int[1] = Saggregate{bound -> @b1to3;  
                         SUPOF(@2, @3);};  
a1to3int[2] = Sarray{SUBOF(@1);  
                     index_type -> !Sinteger;  
                     index_from -> ('1');  
                     SUPOF(@4);};  
a1to3int[3] = unique_elements(SUBOF(@1));  
a1to3int[4] = optional_element{SUBOF(@2)};
```

### A.2.2.16 optional\_element

An ARRAY in which the elements may be optional.

EXPRESS specification:

```
*)  
ENTITY optional_element  
  SUBTYPE OF (Sarray);  
END_ENTITY;  
(*)
```

### A.2.2.17 Sbag

A representation of an EXPRESS BAG.

EXPRESS specification:

```
*)
ENTITY Sbag
  SUBTYPE OF (Saggregate);
END_ENTITY;
(*)
```

EXAMPLE 16 – A data instance representing the EXPRESS code fragment:

BAG OF

EXPRESS-I specification:

```
bany[1] = Sbag{SUBOF(@2)};
bany[2] = Saggregate{bound -> ?;
                     SUPOF(@1)};
```

**A.2.2.18 Slist**

A representation of an EXPRESS LIST.

EXPRESS specification:

```
*)
ENTITY Slist
  SUBTYPE OF (Saggregate);
END_ENTITY;
(*)
```

EXAMPLE 17 – A data instance representing the EXPRESS code fragment:

LIST [2:] OF UNIQUE

EXPRESS-I specification:

```
list2u[1] = Slist{SUBOF(@2)};
list2u[2] = Saggregate{bound -> @b2up; SUPOF(@1, @3)};
list2u[3] = unique_elements{SUBOF(@2)};
b2up = bound{low -> ('2'); high -> ('?')};
```

**A.2.2.19 Xentity**

A representation of an EXPRESS ENTITY.

EXPRESS specification:

```
*)  
ENTITY Xentity;  
    name          : express_id;  
    explicit_attributes : LIST OF UNIQUE explicit_attr;  
    derived_attributes  : LIST OF UNIQUE derived_attr;  
    inverse_attributes   : LIST OF UNIQUE inverse_attr;  
    unique_constraints   : LIST OF UNIQUE uniqueness;  
    where_constraints    : where_clause;  
END_ENTITY;  
(*
```

Attribute definitions:

**name:** The identifier of the entity.  
**explicit\_attributes:** The set of explicit attributes.  
**derived\_attributes:** The set of derived attributes.  
**inverse\_attributes:** The set of inverse attributes.  
**unique\_constraints:** The set of uniqueness constraints.  
**where\_constraints:** The set of domain constraints.

### A.2.2.20 abstract\_supertype

An **Xentity** that is an ABSTRACT SUPERTYPE.

EXPRESS specification:

```
*)  
ENTITY abstract_supertype  
    SUBTYPE OF (Xentity);  
END_ENTITY;  
(*
```

### A.2.2.21 Ssubtype

An **Xentity** that is a SUBTYPE of one or more **Xentity**.

EXPRESS specification:

```
*)  
ENTITY Ssubtype  
    SUBTYPE OF (Xentity);  
    subof : LIST [1:?] OF UNIQUE Xentity;  
END_ENTITY;
```

(\*

Attribute definitions:**subof:** The immediate supertype entities.**A.2.2.22 subtype\_constraint**

A representation of a constraint on the allowable combinations among the subtypes of a supertype.

EXPRESS specification:

```
*)
ENTITY subtype_constraint;
  the_supertype : Xentity;
  constraint    : subtype_expression;
END_ENTITY;
(*)
```

Attribute definitions:**the\_supertype:** The supertype ENTITY applying the constraint to its subtypes.**constraint:** The applied constraint.**A.2.2.23 attribute**

A generalization of an EXPRESS attribute.

EXPRESS specification:

```
*)
ENTITY attribute
  ABSTRACT SUPERTYPE OF (ONEOF(explicit_attr, derived_attr, inverse_attr));
  name      : express_id;
  avalue    : reqopt;
  the_type : type_spec;
END_ENTITY;
(*)
```

Attribute definitions:**name:** The identifier (i.e., role name) of the attribute.**avalue:** Specification of whether or not a conforming instance of the attribute requires a value.**the\_type:** The type of the attribute.

### A.2.2.24 redeclared

An attribute that is a redeclaration of one in a supertype.

#### EXPRESS specification:

```
*)  
ENTITY redeclared  
  SUBTYPE OF (attribute);  
  ancestor : Xentity;  
  base_attr : attribute;  
END_ENTITY;  
(*
```

#### Attribute definitions:

**ancestor:** The supertype which has the original attribute.

**base\_attr:** The original attribute.

### A.2.2.25 explicit\_attr

An explicit attribute.

#### EXPRESS specification:

```
*)  
ENTITY explicit_attr  
  SUBTYPE OF (attribute);  
END_ENTITY;  
(*
```

### A.2.2.26 derived\_attr

A derived attribute.

#### EXPRESS specification:

```
*)  
ENTITY derived_attr  
  SUBTYPE OF (attribute);  
  derivation : expression;  
END_ENTITY;  
(*
```

Attribute definitions:

**derivation:** The specification of how the attribute value is to be calculated.

**A.2.2.27 inverse\_attr**

An inverse attribute.

EXPRESS specification:

```
*)
ENTITY inverse_attr
  SUBTYPE OF (attribute);
  (* next line should be 'attribute\attr_type : Xentity;';
     but fedex hiccups on that valid code *)
  ATTRIBUTE_BACKSLASH_attr_type : Xentity;
    role
      : attribute;
END_ENTITY;
(*)
```

Attribute definitions:

**attr\_type:** The inverse entity.

**role:** The particular attribute in the inverse entity.

**A.2.2.28 uniqueness**

A uniqueness constraint on entities.

EXPRESS specification:

```
*)
ENTITY uniqueness;
  label
    : OPTIONAL express_id;
  (* next line should be: 'entities : BAG [1:?] OF UNIQUE Xentity;';
     but fedex does not understand EXPRESS edition 2 *)
  entities : LIST [1:?] OF UNIQUE Xentity;
END_ENTITY;
(*)
```

Attribute definitions:

**label:** An optional label.

**entities:** The set of entities that shall be mutually unique.

### A.2.2.29 algorithm

A generalization of the EXPRESS algorithms, namely FUNCTION, PROCEDURE, and RULE.

#### EXPRESS specification:

```
*)  
ENTITY algorithm  
  ABSTRACT SUPERTYPE OF (ONEOF(Xfunction, Xprocedure, Xrule));  
  name    : express_id;  
  header  : algorithm_header;  
  body    : algorithm_body;  
END_ENTITY;  
(*
```

#### Attribute definitions:

- name:** The identifier of the algorithm.
- header:** The header of the algorithm.
- body:** The body of the algorithm.

### A.2.2.30 Xfunction

A representation of an EXPRESS FUNCTION.

#### EXPRESS specification:

```
*)  
ENTITY Xfunction  
  SUBTYPE OF (algorithm);  
END_ENTITY;  
(*
```

#### Informal propositions:

- ip1:** The **header** shall conform to the EXPRESS syntax for a function header.
- ip2:** the **body** shall conform to the EXPRESS syntax for a function body.

### A.2.2.31 Xprocedure

A representation of an EXPRESS PROCEDURE.

#### EXPRESS specification:

```
*)
```

```
ENTITY Xprocedure
  SUBTYPE OF (algorithm);
END_ENTITY;
(*)
```

Informal propositions:

**ip1:** The **header** shall conform to the EXPRESS syntax for a procedure header.

**ip2:** the **body** shall conform to the EXPRESS syntax for a procedure body.

### A.2.2.32 Xrule

A representation of an EXPRESS RULE.

EXPRESS specification:

```
*)
ENTITY Xrule
  SUBTYPE OF (algorithm);
END_ENTITY;
(*)
```

Informal propositions:

**ip1:** The **header** shall conform to the EXPRESS syntax for a rule header.

**ip2:** the **body** shall conform to the EXPRESS syntax for a rule body.

### A.2.3 End of schema lexical\_schema

EXPRESS specification:

```
*)
END_SCHEMA; -- end of lexical_schema
(*)
```

### A.3 Schema iconic\_schema

This schema provides a model for the representation of an EXPRESS-G diagram.

EXPRESS specification:

```
*)
SCHEMA iconic_schema;
REFERENCE FROM lexical_schema;
REFERENCE FROM support;
```

(\*

### A.3.1 Type definitions

#### A.3.1.1 gm\_kind

An enumeration of the various kinds of EXPRESS-G models.

EXPRESS specification:

```
*)  
TYPE gm_kind = ENUMERATION OF (schema_complete, schema_partial,  
                                entity_complete, entity_partial);  
END_TYPE;  
(*)
```

Enumerated item definitions:

**schema\_complete:** A complete schema level model.

**schema\_partial:** A partial schema level model.

**entity\_complete:** A complete entity level model.

**entity\_partial:** A partial entity level model.

#### A.3.1.2 Gtype

The various EXPRESS-G types.

EXPRESS specification:

```
*)  
TYPE Gtype = SELECT (Genumeration, Gselect, Gdefined_type, Gsimple_type);  
END_TYPE;  
(*)
```

#### A.3.1.3 any\_type\_or\_schema

A selection between any EXPRESS type or SCHEMA.

EXPRESS specification:

```
*)  
TYPE any_type_or_schema = SELECT (any_type, Xschema);  
END_TYPE;  
(*)
```

### A.3.1.4 line\_style

An enumeration of graphical line styles.

#### EXPRESS specification:

```
*)
TYPE line_style = ENUMERATION OF (weak, normal, strong);
END_TYPE;
(*)
```

#### Enumerated item definitions:

**weak:** A weak (dashed) line.

**normal:** A normal line.

**strong:** A strong (thick) line.

### A.3.1.5 simple\_line\_end

An enumeration of simple line endings.

#### EXPRESS specification:

```
*)
TYPE simple_line_end = ENUMERATION OF (empty, emphasis, import);
END_TYPE;
(*)
```

#### Enumerated item definitions:

**empty:** No graphic.

**emphasis:** A graphic that is an open circle.

**import:** A graphic that is an arrowhead.

### A.3.1.6 line\_end\_style

#### EXPRESS specification:

```
*)
TYPE line_end_style = SELECT(simple_line_end, target, goto);
END_TYPE;
(*)
```

## A.3.2 Entity definitions

### A.3.2.1 iconic\_model

An **iconic\_model** is an information model specified using the EXPRESS-G iconic language. It is composed of one or more sheets.

#### EXPRESS specification:

```
*)  
ENTITY iconic_model;  
    kind      : gm_kind;  
    components : SET [1:?] OF sheet;  
    title     : STRING;  
    administrative : admin_data;  
END_ENTITY;  
(*
```

#### Attribute definitions:

**kind:** The kind of iconic model (e.g., the level and detail).

**components:** The set of sheets comprising the information model.

**title:** The title of the model.

**administrative:** Adminstrative data.

EXAMPLE 18 – A data instance of an **iconic\_model**

#### EXPRESS-I specification:

```
im1 = iconic_model{kind -> !schema_partial;  
                  components -> (@sheet1);  
                  title -> 'partial schema level model title';  
                  administrative -> @admin;};
```

### A.3.2.2 sheet

A **sheet** contains one or more **pages**.

#### EXPRESS specification:

```
*)  
ENTITY sheet;  
    unit   : length_unit;  
    pages : SET OF page;  
DERIVE
```

```

title : STRING := the_model.title;
total_pages : INTEGER := pages_in_model(the_model);
INVERSE
the_model : iconic_model FOR components;
END_ENTITY;
(*

```

Attribute definitions:

**unit:** The unit of length for all pages on the sheet.

**pages:** The set of pages on the sheet.

**title:** The title of the model.

**total\_pages:** The number of pages in the model.

**the\_model:** The **iconic\_model** which uses this sheet.

### A.3.2.3 page

An EXPRESS-G model is composed of icons positioned on one or more **pages**.

EXPRESS specification:

```

*)
ENTITY page;
  id          : INTEGER;
  location    : point;
  size        : width_height;
  schemas     : SET OF Gschema;
  entities    : SET OF Gentity;
  types       : SET OF Gtype;
  lines       : SET OF Gline;
  connectors  : SET OF composition;
  title_loc   : point;
DERIVE
  unit        : length_unit := the_sheet.unit;
  main_title : STRING := make_main_title(the_sheet);
  max_id     : INTEGER := the_sheet.total_pages;
INVERSE
  the_sheet : sheet FOR pages;
WHERE
  limit_id : {1 <= id <= max_id};
END_ENTITY;
(*

```

Attribute definitions:

**id:** An identifying number for the page.

**location:** The location of the bottom left-hand corner of the page on a **sheet**.

**size:** The width and height of the page.

**schemas:** The set of schema icons on the page.

**entities:** The entity icons on the page.

**types:** The set of type icons on the page.

**lines:** The set of relationship lines on the page.

**title\_loc:** The approximate location of the centre of the page title.

**unit:** The unit of measure for the page.

**main\_title:** The general title for the page (i.e., excluding ‘Page x of N’).

**max\_id:** The total number of pages in the model.

**the\_sheet:** The sheet on which the page is placed.

Formal propositions:

**limit\_id:** The value of the **id** shall be one or more and not exceed the total number of pages in the model.

NOTE 1 – Constraints could be added to ensure the distinction between entity level and schema level models, at the cost of model complexity.

#### A.3.2.4 icon

A generalization of an EXPRESS-G icon. The graphical form of an **icon** is described in annex ?? for each subtype.

EXPRESS specification:

```
*)  
ENTITY icon  
ABSTRACT SUPERTYPE OF (ONEOF(Gschema, Gentity, Genumeration,  
                                Gselect, Gdefined_type, Gsimple_type,  
                                composition));  
    centre : point;  
    size   : width_height;  
END_ENTITY;  
(*
```

Attribute definitions:

**centre:** The location of the centre of the icon.

**size:** The width and height of the icon.

### A.3.2.5 Gschema

An icon representing a SCHEMA.

#### EXPRESS specification:

```
*)  
ENTITY Gschema  
  SUBTYPE OF (icon);  
  basis : Xschema;  
END_ENTITY;  
(*
```

#### Attribute definitions:

**basis:** The SCHEMA.

### A.3.2.6 Gentity

An icon representing an ENTITY.

#### EXPRESS specification:

```
*)  
ENTITY Gentity  
  SUBTYPE OF (icon);  
  basis : Xentity;  
END_ENTITY;  
(*
```

#### Attribute definitions:

**basis:** The ENTITY.

### A.3.2.7 Genumeration

An icon representing an ENUMERATION type.

#### EXPRESS specification:

```
*)  
ENTITY Genumeration  
  SUBTYPE OF (icon);  
  basis : Xenumeration;  
END_ENTITY;  
(*
```

Attribute definitions:

**basis:** The ENUMERATION.

### A.3.2.8 Gselect

An icon representing a SELECT type.

EXPRESS specification:

```
*)  
ENTITY Gselect  
  SUBTYPE OF (icon);  
  basis : Xselect;  
END_ENTITY;  
(*
```

Attribute definitions:

**basis:** The SELECT.

### A.3.2.9 Gdefined\_type

An icon representing a TYPE that is neither an ENUMERATION nor a SELECT type.

EXPRESS specification:

```
*)  
ENTITY Gdefined_type  
  SUBTYPE OF (icon);  
  basis : defined_type;  
END_ENTITY;  
(*
```

Attribute definitions:

**basis:** The defined type.

### A.3.2.10 Gsimple\_type

An icon representing a simple type.

EXPRESS specification:

```
*)  
ENTITY Gsimple_type
```

```

SUBTYPE OF (icon);
basis : simple_type;
END_ENTITY;
(*

```

Attribute definitions:

**basis:** The simple type.

### A.3.2.11 composition

An icon that represents an EXPRESS-G composition symbol.

EXPRESS specification:

```

*)
ENTITY composition
  SUBTYPE OF (icon);
END_ENTITY;
(*

```

### A.3.2.12 target

An icon that represents an EXPRESS-G reference onto a page.

EXPRESS specification:

```

*)
ENTITY target
  SUBTYPE OF (composition);
  refid : INTEGER;
  from_pages : SET OF page;
END_ENTITY;
(*

```

Attribute definitions:

**refid:** The reference number.

**from\_pages:** The set of pages that refer to this composition icon.

### A.3.2.13 goto

An icon that represents an EXPRESS-G reference onto another page.

EXPRESS specification:

```
*)  
ENTITY goto  
  SUBTYPE OF (composition);  
  to_ref : target;  
  to_type : any_type_or_schema;  
END_ENTITY;  
(*
```

Attribute definitions:

**to\_ref:** The target of this reference.

**to\_type:** The type or schema being referred to.

### A.3.2.14 use\_import

A composition icon that represents a definition USED from another SCHEMA.

EXPRESS specification:

```
*)  
ENTITY use_import  
  SUBTYPE OF (composition);  
  origin : Xschema;  
  definition : Xentity;  
  aka : OPTIONAL express_id;  
END_ENTITY;  
(*
```

Attribute definitions:

**origin:** The exporting SCHEMA.

**definition:** The imported ENTITY.

**aka:** The alias of the ENTITY in the importing SCHEMA.

### A.3.2.15 ref\_import

A composition icon that represents a definition REFERENCED from another SCHEMA.

EXPRESS specification:

```
*)  
ENTITY ref_import  
  SUBTYPE OF (composition);  
  origin : Xschema;
```

```

definition : any_type;
aka        : OPTIONAL express_id;
END_ENTITY;
(*

```

Attribute definitions:

**origin:** The exporting SCHEMA.

**definition:** The imported type.

**aka:** The alias of the definition in the importing SCHEMA.

### A.3.2.16 g\_relationship

A logical relationship between elements of an EXPRESS-G model.

EXPRESS specification:

```

*)
ENTITY g_relationship;
  from_element : OPTIONAL any_type_or_schema;
  to_element   : OPTIONAL any_type_or_schema;
  connector    : SET OF Gline;
END_ENTITY;
(*

```

Attribute definitions:

**from\_element:** The logical starting point of the relationship.

**to\_element:** The logical end of the relationship.

**connector:** The graphical representation of the relationship.

### A.3.2.17 Gline

A curve in a two-dimensional space.

EXPRESS specification:

```

*)
ENTITY Gline;
  space      : page;
  start     : point;
  intermediates : LIST OF UNIQUE point;
  finish    : point;
  style     : line_style;
  start_graphic : line_end_style;

```

```
end_graphic   : line_end_style;
annotations    : SET OF annotation;
END_ENTITY;
(*)
```

Attribute definitions:

**space:** The page on which the curve is drawn.

**start:** The line starts at or near this point.

**intermediates:** The line passes through or near these points.

**finish:** The line ends at or near this point.

**style:** The graphical style of the curve.

**start\_graphic:** The style of the graphic at the start of the curve.

**end\_graphic:** The style of the graphic at the end of the curve.

### A.3.2.18 annotation

Annotation for a line.

EXPRESS specification:

```
*)
ENTITY annotation;
  location : point;
  text      : STRING;
END_ENTITY;
(*)
```

Attribute definitions:

**location:** The annotation text is centred near this point.

**text:** The text for the annotation.

### A.3.3 Function definitions

#### A.3.3.1 make\_main\_title

Generate the text string for the main title for an EXPRESS-G model.

EXPRESS specification:

```
*)
```

```

FUNCTION make_main_title(par : sheet) : STRING;
LOCAL
  result : STRING;
END_LOCAL;
CASE par.the_model.kind OF
  schema_complete : result := 'Complete schema level diagram of ';
  schema_partial : result := 'Partial schema level diagram of ';
  entity_complete : result := 'Complete entity level diagram of ';
  entity_partial : result := 'Partial entity level diagram of ';
END_CASE;
result := result + par.the_model.title;
RETURN(result);
END_FUNCTION;
(*

```

### A.3.3.2 pages\_in\_model

Calculate the number of pages in an iconic model.

#### EXPRESS specification:

```

*)
FUNCTION pages_in_model(par : iconic_model) : INTEGER;
LOCAL
  result : INTEGER := 0;
  number_of_sheets : INTEGER := SIZEOF(par.components);
END_LOCAL;

REPEAT i := 0 TO number_of_sheets;
  result := result + SIZEOF(par.components[i].pages);
END_REPEAT;

RETURN(result);
END_FUNCTION;
(*

```

### A.3.4 End of schema iconic\_schema

#### EXPRESS specification:

```

*)
END_SCHEMA; -- end of iconic_schema;
(*

```

## A.4 Schema support

This schema contains general supporting definitions for the major schemas in this model.

EXPRESS specification:

```
*)  
SCHEMA support;  
(*
```

### A.4.1 Type definitions

#### A.4.1.1 express\_id

An **express\_id** is a representation of an EXPRESS simple identifier.

EXPRESS specification:

```
*)  
TYPE express_id = STRING;  
END_TYPE;  
(*
```

#### A.4.1.2 reqopt

An enumeration of required and optional values.

EXPRESS specification:

```
*)  
TYPE reqopt = ENUMERATION OF (reqval, optval);  
END_TYPE;  
(*
```

#### A.4.1.3 algorithm\_head

A list of strings representing the contents of the signature of an EXPRESS algorithm (e.g., a FUNCTION, PROCEDURE, or RULE).

EXPRESS specification:

```
*)  
TYPE algorithm_head = LIST OF STRING;  
END_TYPE;  
(*
```

#### EXAMPLES

19 – A data instance of an **algorithm\_head** for the EXPRESS code fragment:

```
-- FUNCTION fun
```

```
(par1 : INTEGER; par2 : LIST OF REAL) : LOGICAL;
```

EXPRESS-I specification:

```
algorithm_head{('(par1 : INTEGER; par2 : LIST OF REAL)',  
': LOGICAL;')}
```

20 – A data instance of an **algorithm\_head** for the EXPRESS code fragment:

```
-- RULE a_rule FOR  
(entity_1, entity_2);
```

EXPRESS-I specification:

```
algorithm_head{('(entity1, entity2);')}
```

#### A.4.1.4 algorithm\_body

A list of strings representing the contents of the body of an EXPRESS algorithm (e.g., a FUNCTION, PROCEDURE, or RULE).

EXPRESS specification:

```
*)  
TYPE algorithm_body = LIST [1:?] OF STRING;  
END_TYPE;  
(*
```

#### EXAMPLES

21 – An **algorithm\_body** for a body of a FUNCTION like:

```
-- FUNCTION fun (par : INTEGER) : INTEGER;  
LOCAL  
...  
RETURN(result);  
END_FUNCTION;
```

EXPRESS-I specification:

```
algorithm_body{('LOCAL', ... 'RETURN(result);')}
```

22 – An **algorithm\_body** for a body of a RULE like:

```
-- RULE a_rule FOR (e1, e2);
```

```
LOCAL  
...  
WHERE  
...  
r3 : a.b <> c;  
END_RULE;
```

EXPRESS-I specification:

```
algorithm_body{('LOCAL', ... 'WHERE', ... 'r3 : a.b <> c;')}
```

#### A.4.1.5 where\_clause

A representation of a set of EXPRESS domain rules, as used in a WHERE clause.

EXPRESS specification:

```
*)  
TYPE where_clause = LIST OF STRING;  
END_TYPE;  
(*
```

#### A.4.1.6 expression

A representation of an EXPRESS expression.

EXPRESS specification:

```
*)  
TYPE expression = LIST [1:?] OF STRING;  
END_TYPE;  
(*)
```

EXAMPLE 23 – Some instances of **expression**:

EXPRESS-I specification:

```
expression{('42')}  
expression{('?')}  
expression{('gender = female AND age >= 18'),  
          'XOR',  
          ('gender = male AND age >= 21'))}
```

### A.4.1.7 subtype\_expression

A representation of a subtype constraint expression.

EXPRESS specification:

```
*)
TYPE subtype_expression = LIST [1:?] OF STRING;
END_TYPE;
(*)
```

### A.4.1.8 length\_unit

An enumeration of units of length in the context of printing and drawing on paper.

EXPRESS specification:

```
*)
TYPE length_unit = ENUMERATION OF (mm, cm, inch,
                                     pt, bpt, pc);
END_TYPE;
(*)
```

Enumerated item definitions:

**mm:** Millimeter.

**cm:** Centimeter ( $1\text{cm} = 10\text{mm}$ ).

**inch:** Inch ( $1\text{inch} = 25.4\text{mm}$ ).

**pt:** Point (a typographer's measure) ( $72.27\text{pt} = 1\text{in}$ ).

**bpt:** Big point (a measure used in Postscript) ( $72\text{bpt} = 1\text{in}$ ).

**pc:** Pica (a typographer's measure) ( $1\text{pc} = 12\text{pt}$ ).

### A.4.1.9 point

A location defined in two dimensional space with the first value being the horizontal x coordinate and the second value being the vertical y coordinate.

EXPRESS specification:

```
*)
TYPE point = ARRAY [1:2] OF NUMBER;
END_TYPE;
(*)
```

#### A.4.1.10 alength

A value specifying a length.

##### EXPRESS specification:

```
*)  
TYPE alength = NUMBER;  
WHERE  
  non_negative : SELF >= 0;  
END_TYPE;  
(*
```

#### A.4.1.11 width\_height

A pair of lengths giving the width and height of a rectangular region.

##### EXPRESS specification:

```
*)  
TYPE width_height = ARRAY [1:2] OF alength;  
END_TYPE;  
(*
```

EXAMPLE 24 – An instance of a **width\_height**:

##### EXPRESS-I specification:

```
width_height{[8.5, 11.0]}
```

### A.4.2 Entity definitions

#### A.4.2.1 admin\_data

Data of an administrative nature.

##### EXPRESS specification:

```
*)  
ENTITY admin_data;  
  attr : STRING;  
END_ENTITY;  
(*
```

Attribute definitions:

**attr:** This is just a placeholder at the moment.

### A.4.3 End of schema support

EXPRESS specification:

```
*)  
END_SCHEMA; -- end of support  
(*
```

## Annex B (informative)

### (Incomplete draft of) Example data transfer model data instances

This annex presents a short example of data instances according to the data transfer model specified in annex A. The EXPRESS-I syntax is used for displaying the data instances. The focus of the example is the graphical layout of an ExpressG model; that is, data instances representative of the **iconic\_schema**. The data exhibited here correspond to the EXPRESS-G model shown in figure A.1.

The data instances exhibited do not comprise a complete data set in that much of the data required for the **lexical\_schema** is not shown. A conforming data set would include this data as it is required in order to support the **iconic\_schema** data. In this case, though, it is practically irrelevant for the purposes at hand.

#### B.1 Model example\_data\_transfer

The start of the data.

##### EXPRESS-I specification:

```
*)  
MODEL example_data_transfer;  
(*
```

##### B.1.1 Schema main data

The data instances for **main** schema.

##### EXPRESS-I specification:

```
*)  
SCHEMA_DATA main;  
(*
```

##### B.1.1.1 Data instances

In this example there is only a single entity instance.

##### EXPRESS-I specification:

```
*)  
mv = model_views{lexical -> @lm;  
                 iconic -> (@im);
```

```
    administrative -> @admin1;};  
(*
```

### B.1.1.2 End of main data

EXPRESS-I specification:

```
*)  
END_SCHEMA_DATA; -- main  
(*)
```

### B.1.2 Schema iconic\_schema data

The data instances for the **iconic\_schema**.

EXPRESS-I specification:

```
*)  
SCHEMA_DATA iconic_schema;  
(*)
```

### B.1.3 Data instances

In this example there is a single instance of an **iconic\_model**.

EXPRESS-I specification:

```
*)  
im = iconic_model{kind -> !schema_complete;  
                  components -> (@sheet1);  
                  title = 'a model for the transfer of EXPRESS-G  
                           model representations.';  
                  administrative -> @admin3;  
(*)
```

The sheet(s) comprising the EXPRESS-G model representation.

EXPRESS-I specification:

```
*)  
sheet1 = sheet{unit -> !mm;  
                 pages -> (@page1);  
                 title <- 'a model for the transfer of EXPRESS-G  
                           model representations.';  
                 the_model <- @im;};  
(*)
```

The page(s) comprising the sheet(s).

EXPRESS-I specification:

```
*)
page1 = page{id -> 1;
              location -> [0,0];
              size -> [160, 90];
              schemas -> (@gmain_sch, @lexical_sch, @iconic_sch,
                            @support_sch);
              entities -> ();
              types -> ();
              lines -> (@m2s, @m2l, @m2i, @l2i, @l2s, @i2s,
                          @ladnote, @lxcsnote, @lshnote);
              connectors -> ();
              title_loc -> [80,2]
              unit <- !mm;
              partial_title <- 'a model for the transfer of EXPRESS-G
                                model representations.';
              the_sheet <- @sheet1;};

(*
```

The schema icons.

EXPRESS-I specification:

```
*)
gmain_sch[1] = icon{centre -> [80,50];
                     size -> @wh30by10;
                     SUPOF(@2);};
gmain_sch[2] = Gschema{SUBOF(@1);
                     basis -> @main_sch;};

lexical_sch[1] = icon{centre -> [15,80];
                      size -> @wh30by10;
                      SUPOF(@2);};
lexical_sch[2] = Gschema{SUBOF(@1);
                     basis -> @lexical_sch;};

giconic_sch[1] = icon{centre -> [145,80];
                      size -> @wh30by10;
                      SUPOF(@2);};
giconic_sch[2] = Gschema{SUBOF(@1);
                     basis -> @iconic_sch;};

gsupport_sch[1] = icon{centre -> [80,20];
                      size -> @wh30by10;
                      SUPOF(@2);};
gsupport_sch[2] = Gschema{SUBOF(@1);
                     basis -> @support_sch;};

(*)
```

The logical relationships.

EXPRESS-I specification:

```

*)
MandS = g_relationship{one -> @gmain_sch;
                      another -> @gsupport_sch;
                      connector -> (@m2s, @ladnote);};

MandL = g_relationship{one -> @gmain_sch;
                      another -> @lexical_sch;
                      connector -> (@m2l, @lmnote);};

MandI = g_relationship{one -> @gmain_sch;
                      another -> @giconic_sch;
                      connector -> (@m2i, @limnote);};

LandI = g_relationship{one -> @lexical_sch;
                      another -> @giconic_sch;
                      connector -> (@l2i);};

LandS = g_relationship{one -> @lexical_sch;
                      another -> @gsupport_sch;
                      connector -> (@l2s);};

IandS = g_relationship{one -> @giconic_sch;
                      another -> @gsupport_sch;
                      connector -> (@i2s);};
(*

```

The relationship lines and annotation.

EXPRESS-I specification:

```

*)
-- main to support
m2s = Gline{start -> [80,45];
            intermediates -> ();
            finish -> [80,25];
            style -> !weak;
            start_graphic -> !empty;
            end_graphic -> !emphasis;
            annotations -> ();
            space <- @page1;};

-- main to lexical
m2l = Gline{start -> [72.5,55];
            intermediates -> ([72.5,77.5]);
            finish -> [30,77.5];
            style -> !normal;
            start_graphic -> !empty;
            end_graphic -> !emphasis;
            annotations -> ();
            space <- @page1;};

```

```

-- main to iconic
m2i = Gline{start -> [87.5,55];
             intermediates -> ([87.5,77.5]);
             finish -> [130,77.5];
             style -> !weak;
             start_graphic -> !empty;
             end_graphic -> !emphasis;
             annotations -> ();
             space <- @page1;};

-- lexical and iconic
l2i = Gline{start -> [30,82.5];
             intermediates -> ();
             finish -> [130,82.5];
             style -> !normal;
             start_graphic -> !emphasis;
             end_graphic -> !empty;
             annotations -> ();
             space <- @page1;};

-- lexical to support
l2s = Gline{start -> [15,75];
             intermediates -> ([15,20]);
             finish -> [65,20];
             style -> !weak;
             start_graphic -> !empty;
             end_graphic -> !emphasis;
             annotations -> ();
             space <- @page1;};

-- iconic to support
i2s = Gline{start -> [130,75];
             intermediates -> ([130,20]);
             finish -> [95,20];
             style -> !weak;
             start_graphic -> !empty;
             end_graphic -> !emphasis;
             annotations -> ();
             space <- @page1;};

-- admin_data annotation
ladnote = Gline{start -> [80,35];
                 intermediates -> ();
                 finish -> [100,35];
                 style -> !normal;
                 start_graphic -> !arrowhead;
                 end_graphic -> !empty;
                 annotations -> (@adnote);
                 space <- @page1;};
adnote = annotation{location -> [90,37];
                     text -> 'admin_data';};

```

```

-- lexical_model annotation
lmnote = Gline{start -> [72.5,55];
               intermediates -> ();
               finish -> [52.5,55];
               style -> !normal;
               start_graphic -> !arrowhead;
               end_graphic -> !empty;
               annotations -> (@lmnote);
               space <- @page1;};
lmnote = annotation{location -> [62.5,57];
                     text -> 'lexical_model';};

-- iconic_model annotation
imnote = Gline{start -> [87.5,55];
               intermediates -> ();
               finish -> [107.5,55];
               style -> !medium;
               start_graphic -> !arrowhead;
               end_graphic -> !empty;
               annotations -> (@imnote);
               space <- @page1;};
imnote = annotation{location -> [97.5,57];
                     text -> 'iconic_model';};
(*

```

### B.1.3.1 End of iconic\_schema data

EXPRESS-I specification:

```

*)
END_SCHEMA_DATA; -- iconic_schema
(*

```

## B.1.4 Schema lexical\_schema data

The data instances for the **lexical\_schema**.

EXPRESS-I specification:

```

*)
SCHEMA_DATA lexical_schema;
(*

```

### B.1.4.1 Data instances

The **lexical\_model** instance.

EXPRESS-I specification:

```
*)
lm = lexical_model{components -> (@main_sch, @lexical_sch, @iconic_sch,
                                @support_sch);
                    administrative -> @admin2;};

(*)
```

The entity instances for the four schemas.

EXPRESS-I specification:

```
*)
main_sch = Xschema{name ->      'main';
                     uses ->      (@use_lexical, @use_iconic);
                     references -> (@ref_admin);
                     constants -> ();
                     types ->      ();
                     entities ->  (@mvs);
                     rules ->      ();
                     functions -> ();
                     procedures -> ()};

lexical_sch = Xschema{name ->    'lexical_schema';
                     uses ->      ();
                     references -> (@ref_all_support);
                     constants -> ();
                     types ->      (* 3 of these *);
                     entities ->  (* many of these *);
                     rules ->      ();
                     functions -> ();
                     procedures -> ()};

iconic_sch = Xschema{name ->    'iconic_schema';
                     uses ->      ();
                     references -> (@ref_all_lexical, @ref_all_support);
                     constants -> ();
                     types ->      (* 6 of these *);
                     entities ->  (* many of these *);
                     rules ->      ();
                     functions ->  (* 2 of these *);
                     procedures -> ()};

support_sch = Xschema{name ->   'support';
                     uses ->      ();
                     references -> ();
                     constants -> ();
                     types ->      (* many of these *);
                     entities ->  (* 1 of these *);
                     rules ->      ();
                     functions -> ();
                     procedures -> ()};
```

(\*

The instance data for the USE and REFERENCE statements.

EXPRESS-I specification:

```
*)
ref_all_lexical = Xreference{source -> @lexical_sch;
                             imports -> ();};
ref_all_support = Xreference{source -> @support_sch;
                            imports -> ());}

ref_admin = Xreference{source -> @support_sch;
                      imports -> (@admin);};
admin = perhaps_renamed{original -> @Admin_data; aka -> ?;};

use_lexical = Xuse{source -> @lexical_sch;
                   imports -> (@aka_lex_mod);};
aka_lex_mod = perhaps_renamed{original -> @lex_mod; aka -> ?;};

use-iconic = Xuse{source -> @iconic_sch;
                  imports -> (@aka_icon_mod);};
aka_icon_mod = perhaps_renamed{source -> @icon_mod; aka -> ?;};
(*
```

There are many more data instances required for completion of the lexical model. As the main purpose of this example is to exhibit data instances for the representation of an EXPRESS-G schema level model, most of this data is of peripheral interest. Therefore, only a data instance for the **model\_views** ENTITY is given.

The instance data for the **model\_views** ENTITY declaration.

EXPRESS-I specification:

```
*)
mvs = Xentity{name -> 'model_views';
               explicit_attributes -> (@lex_att, @icon_att, @admin_att);
               derived_attributes -> ();
               inverse_attributes -> ();
               unique_constraints -> ();
               where_constraints -> ()};

lex_att[1] = attribute{name -> 'lexical';
                       avalue -> !reqval;
                       the_type -> @lex_type;
                       SUPOF(@2);};
lex_att[2] = explicit_attr{SUBOF(@1);};
lex_type = type_spec{aggregation -> ();
                     base -> @lex_mod;};

icon_att[1] = attribute{name -> 'iconic';
                        avalue -> !reqval;}
```

```
        the_type -> @set1p_of_im;
        SUPOF(@2);};

icon_att[2] = explicit_attr{SUBOF(@1);};
set1p_of_im = type_spec{aggregation -> ();
                        base -> @icon_mod;};

set1p[1] = Saggregate{bound -> @b1p;
                      SUPOF(@2, @3);};
set1p[2] = Sbag{SUBOF(@1);};
set1p[3] = unique_elements{SUBOF(@1);};

b1p = bound_spec{low -> '1';
                  high -> '?';};

admin_att[1] = attribute{name -> 'administrative';
                           avalue -> !reqval;
                           the_type -> @admin_type;
                           SUPOF(@2);};
admin_att[2] = explicit_attr{SUBOF(@1);};
(*
```

#### B.1.4.2 End of lexical\_schema data

EXPRESS-I specification:

```
*)  
END_SCHEMA_DATA; -- lexical_schema  
(*)
```

#### B.1.5 Schema support data

The data instances for the **support** schema.

EXPRESS-I specification:

```
*)  
SCHEMA_DATA support;  
(*)
```

#### B.1.5.1 Data instances

One instance of the **width\_height** defined type.

EXPRESS-I specification:

```
*)  
wh30by10 = width_height{[30, 10]};  
(*)
```

The instances of **admin\_data**.

EXPRESS-I specification:

```
*)
admin1 = admin_data{attr -> 'admin data 1';}
admin2 = admin_data{attr -> 'admin data 2';}
admin3 = admin_data{attr -> 'admin data 3';}
(*)
```

### B.1.6 End of support data

EXPRESS-I specification:

```
*)
END_SCHEMA_DATA; -- support
(*)
```

### B.1.7 End of model example\_data\_transfer

EXPRESS-I specification:

```
*)
END_MODEL; -- example_data_transfer
(*)
```